Notes

The project is conducted with a Jetson Orin Nano Developer Kit 8GB. It originally started with Jetson firmware 4.1 causing a different boot up than the one recommended.

1.1 Recommended First Boot: Flashing Jetpack Jetson Orin Nano Developer Kit Using Ethcher

This will depend on on what firmware version in on ther Jetson Board.

To prepare your microSD card, you'll need a computer with Internet connection and the ability to read and write SD cards, either via a built-in SD card slot or adapter.

1. Download the Jetson Orin Nano Developer Kit SD Card image from JetPack SDK Page , and note where it was saved on the computer.

- **Before writing the sd card go to the disc manager and ensure that there are no pations on the sd card you will not need to worry about this with a new sd card**

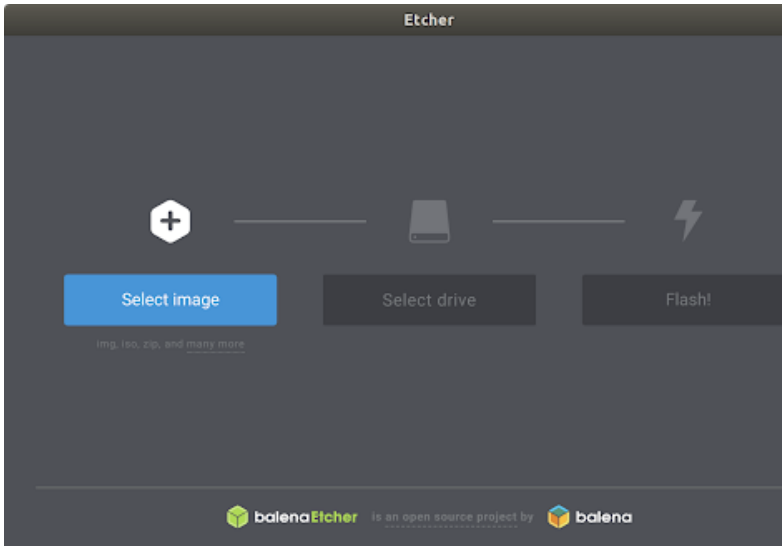
2. Write the image to your microSD card by following the instructions here according to the type of computer you are using: Windows, Mac, or Linux.

Use the following instructions to download Ether

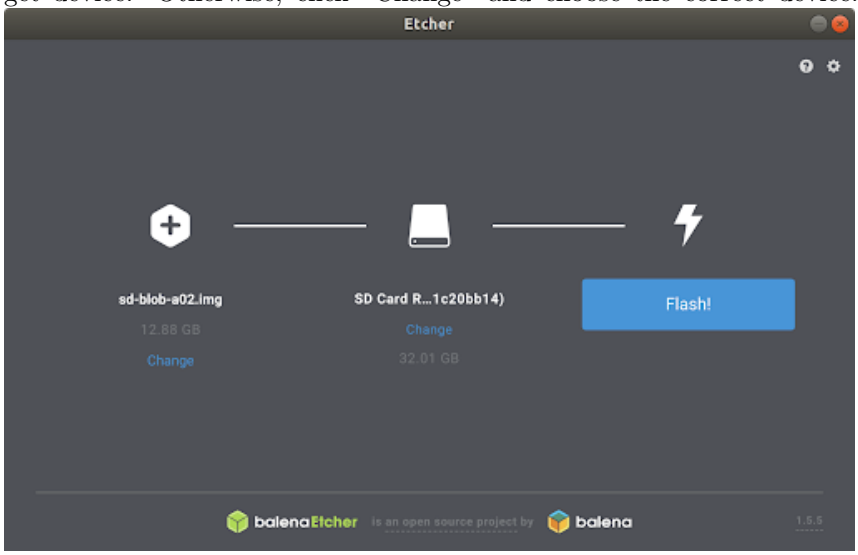
Instructions for Linux

You can either write the SD card image using a graphical program like Etcher, or via command line.

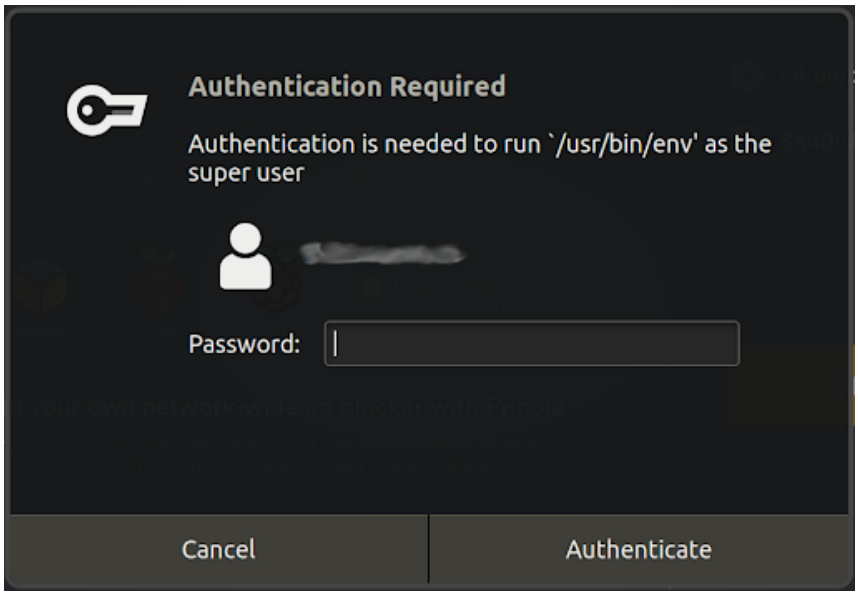
Etcher Instructions



1. Download, install, and launch Etcher.
2. Click "Select image" and choose the zipped image file downloaded earlier.
3. Insert your microSD card. If you have no other external drives attached, Etcher will automatically select the microSD card as target device. Otherwise, click "Change" and choose the correct device.

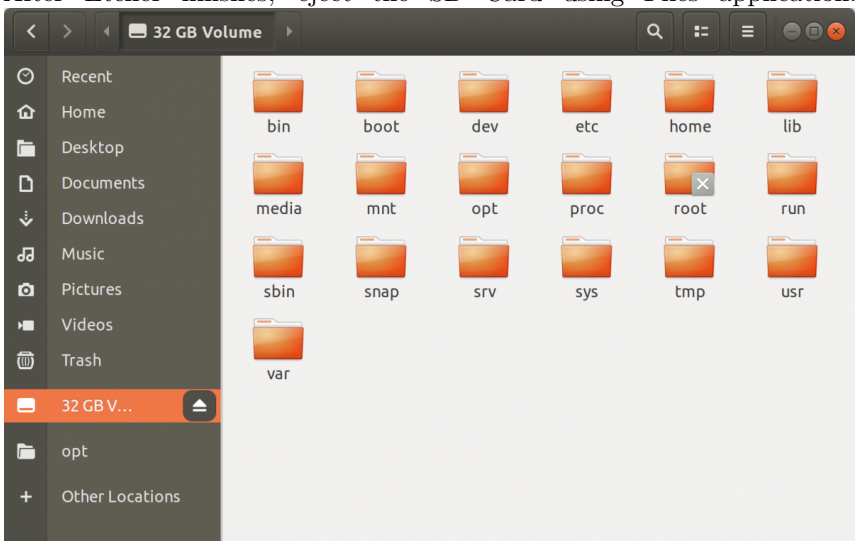


4. Click "Flash!" Your OS may prompt for your username and password before



it allows Etcher to proceed. It will take Etcher 10-15 minutes to write and validate the image if your microSD card is connected via USB3.

5. After Etcher finishes, eject the SD Card using Files application:



6. Physically remove microSD card from the computer.

Command Line Instructions

1. Open the Terminal application by pressing `Ctrl + Alt + t`.
2. Insert your microSD card, then use a command like this to show which

disk device was assigned to it:

```
dmesg | tail | awk '$2 == "sd" {print}'
```

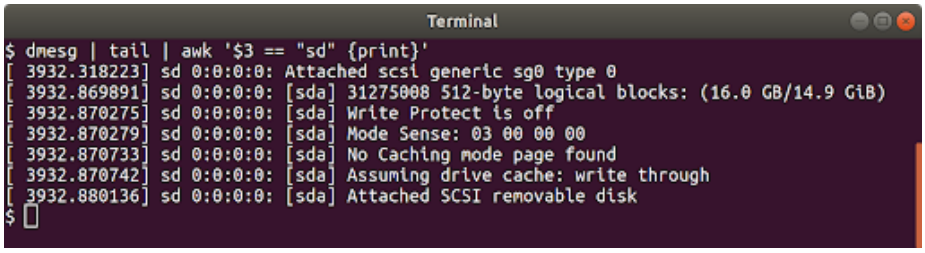


Figure 1: alt text

3. Use this command to write the zipped SD card image to the microSD card:

```
/usr/bin/unzip -p ~/Downloads/jp60-orin-nano-sd-card-image.zip |
sudo /bin/dd of=/dev/sda bs=1M status=progress
```

For example:

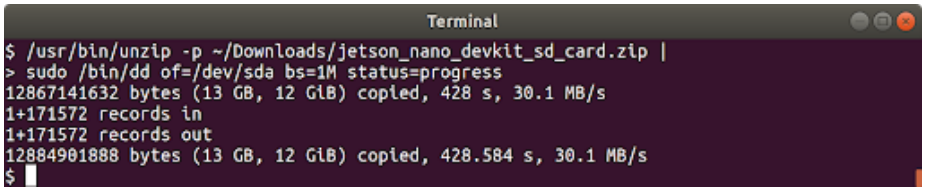


Figure 2: alt text

When the dd command finishes, eject the disk device from the command line: `sudo eject /dev/sda`

4. Physically remove microSD card from the computer.

After your microSD card is ready, proceed to Setup your developer kit.

If this doesn't work use the Jetpack 5.1.3 instead of Jetpack 6.0. Use the following link

Setup and First Boot

Setup Steps 1. Insert the microSD card (with system image already written to it) into the slot on the underside of the Jetson Orin Nano module. 2. Power on your computer display and connect it. 3. Connect the USB keyboard and mouse. 4. Connect the provided power supply. The Jetson Orin Nano Developer Kit will power on and boot automatically.

First Boot

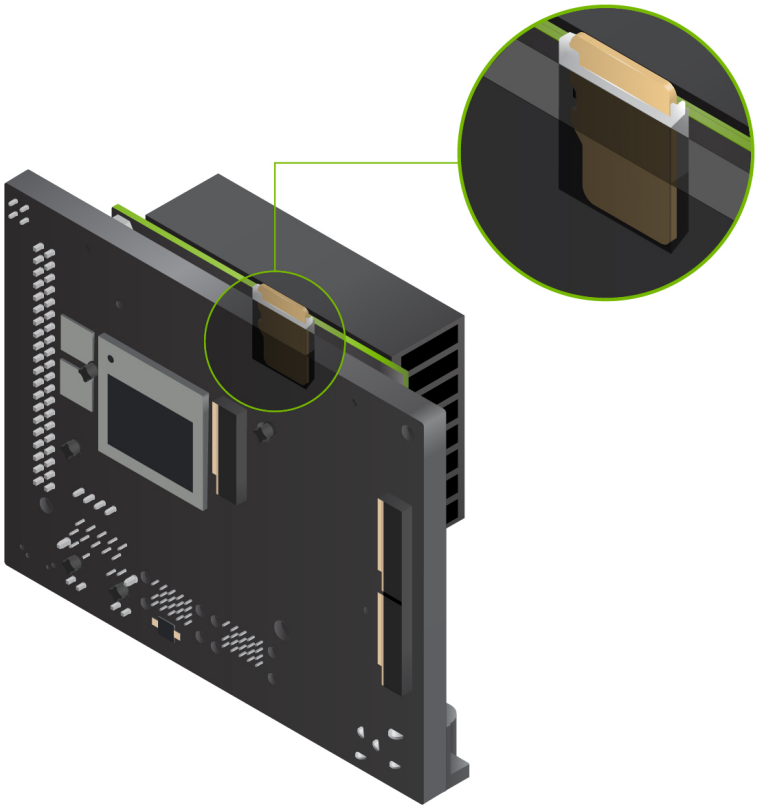


Figure 3: alt text

A green LED next to the USB-C connector will light as soon as the developer kit powers on. When you boot the first time, the Jetson Orin Nano Developer Kit will take you through some initial setup, including:

- Review and accept NVIDIA Jetson software EULA
- Select system language, keyboard layout, and time zone
- Connect to Wireless network
- Create username, password, and computer name
- Log in

After Logging In You will see this screen. Congratulations!

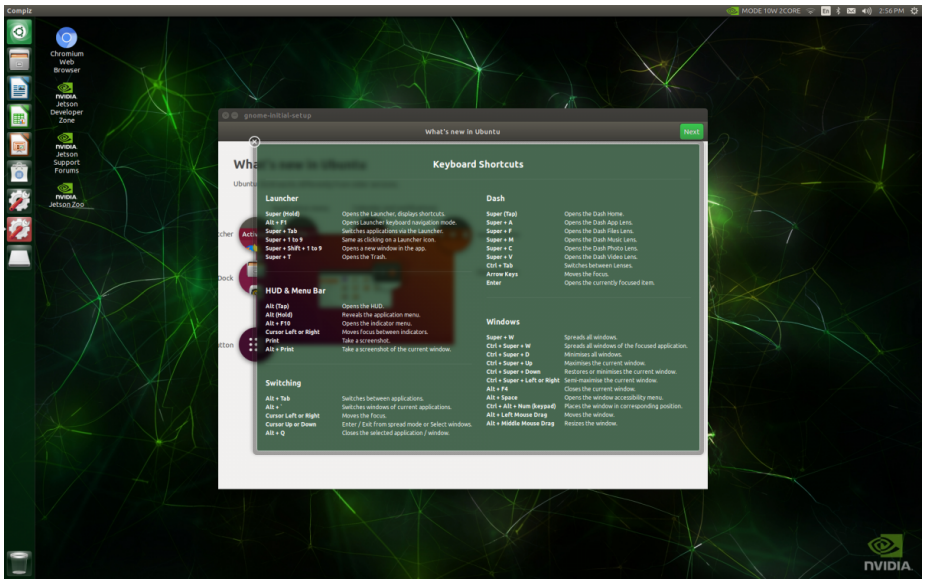


Figure 4: alt text

1.2 Flash to Jetpack +6.0 If Software Won't Update From 4.1 -> 5.1.3 -> +6.0

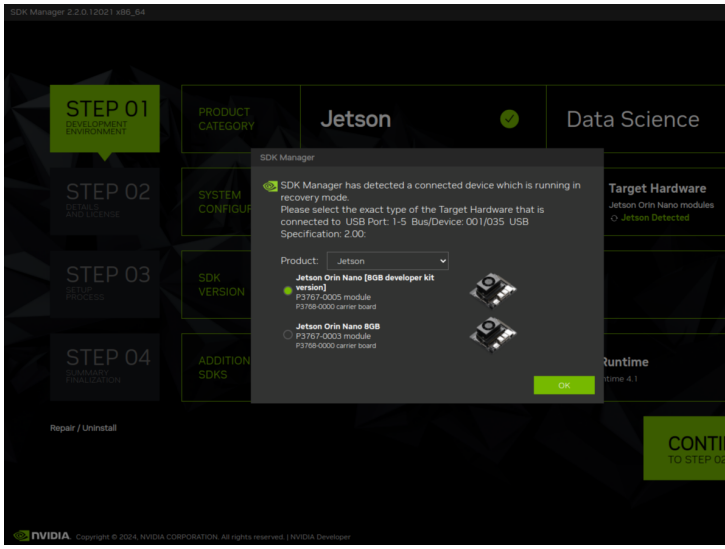
Equitment needed

- Jetson Orin Nano Dev Kit
- Host PC running on **Unbuntu 20.04** with **Nvidia SDK Manager 2.2.0**
- 128 gb sc card
- 500 gb NVMe
- Nvidia Jetson Power cable (19.0V, 2.37A)
- Data capable usb to type-c
- Jumper wire
- Display with display port

- Display port cable
- Keyboard
- Mouse

Hareware Setup

1. Install blank sd card and blank NVMe
2. Connect juper cable to put the Jetson in Force Recovery mode. (This enbles the board to be detected by the SDK manager)
3. Connect the usb to host computer and the other end into the type-c port on the Jetson.
4. Connect the power source to the Jetson.
5. Open the SDK Manager (2.2.0 version) on the Host computer (Ubuntu 20.04)



6. Select The Jetson Orin Nano Developer Kit

Flashing Jetpack 5.1.3

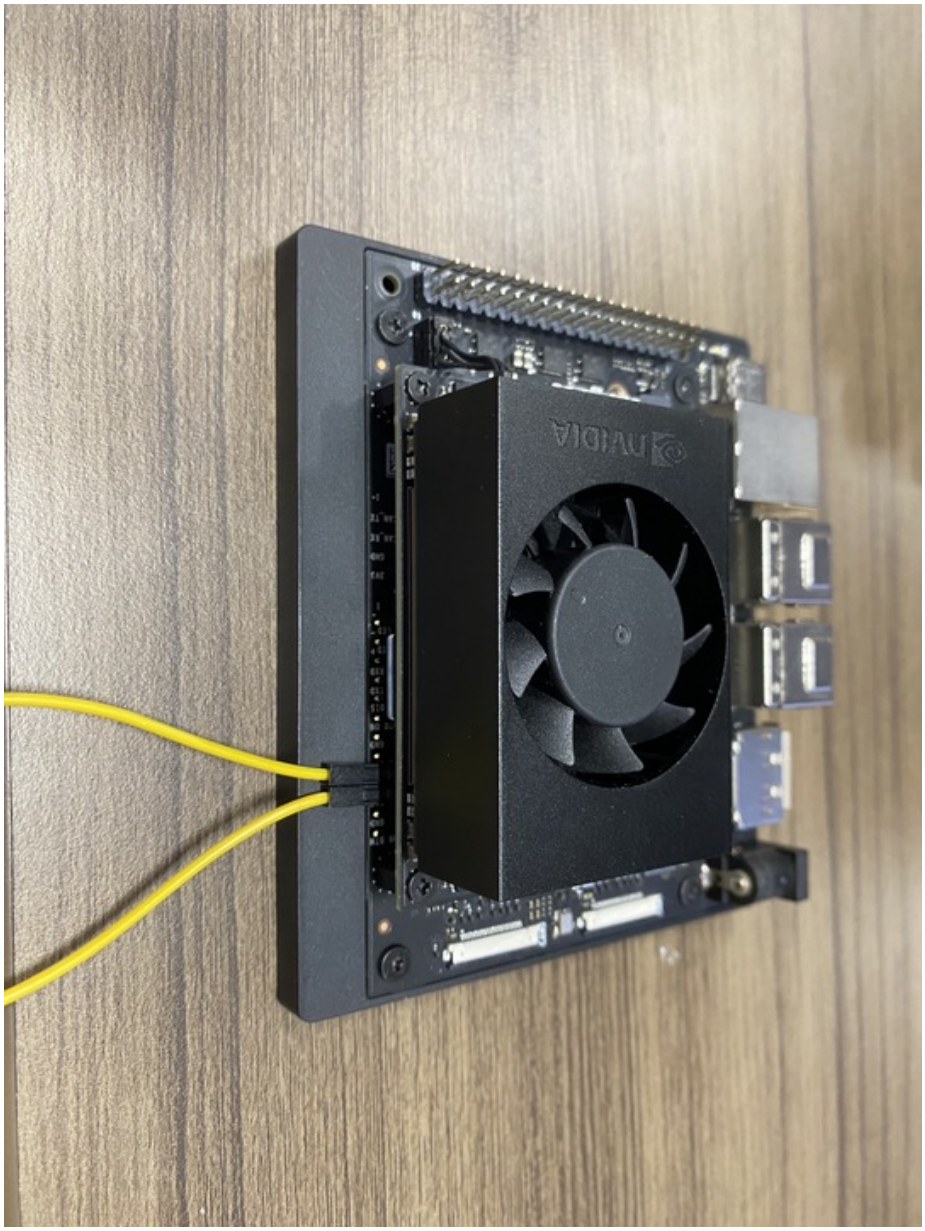


Figure 5: alt text

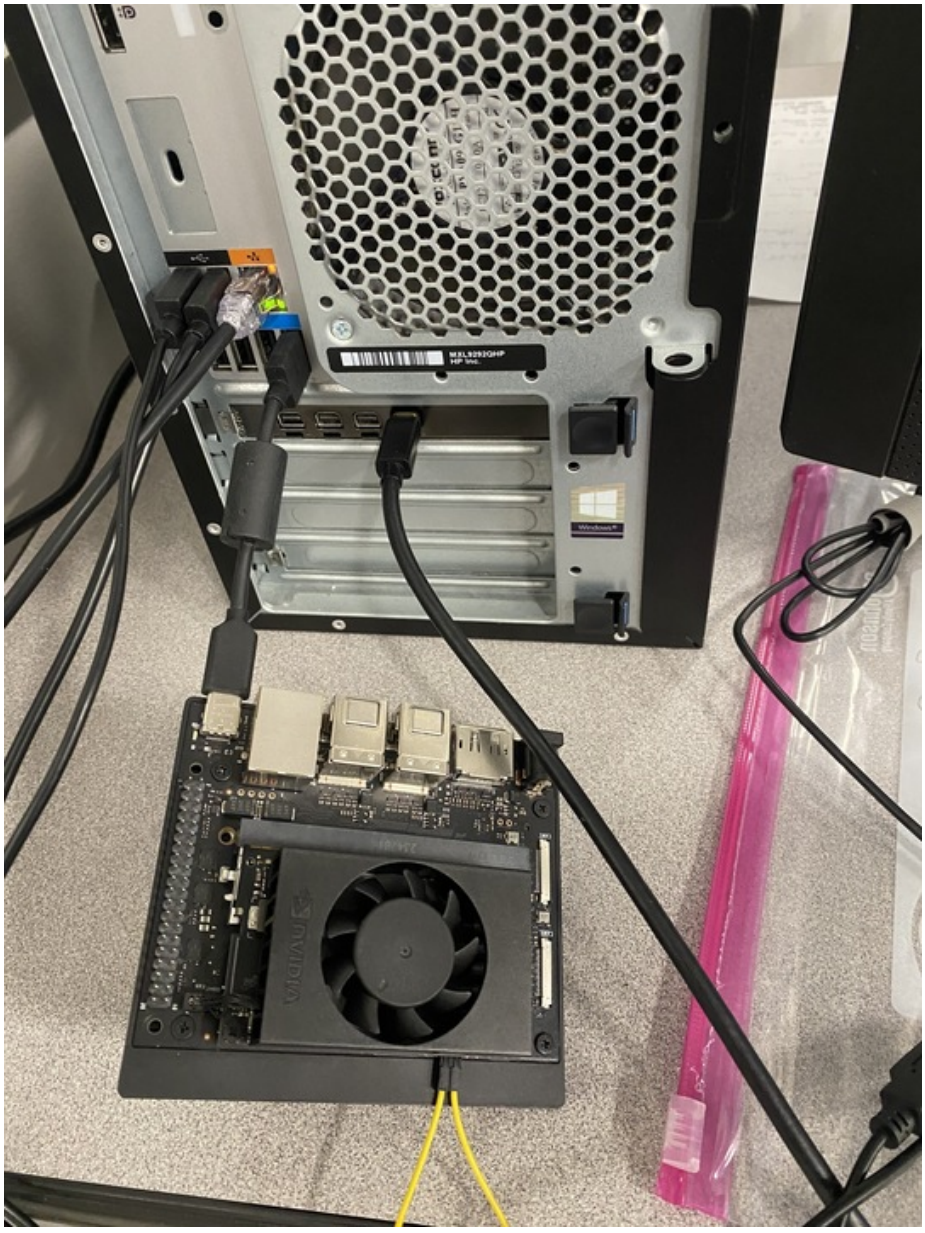


Figure 6: alt text

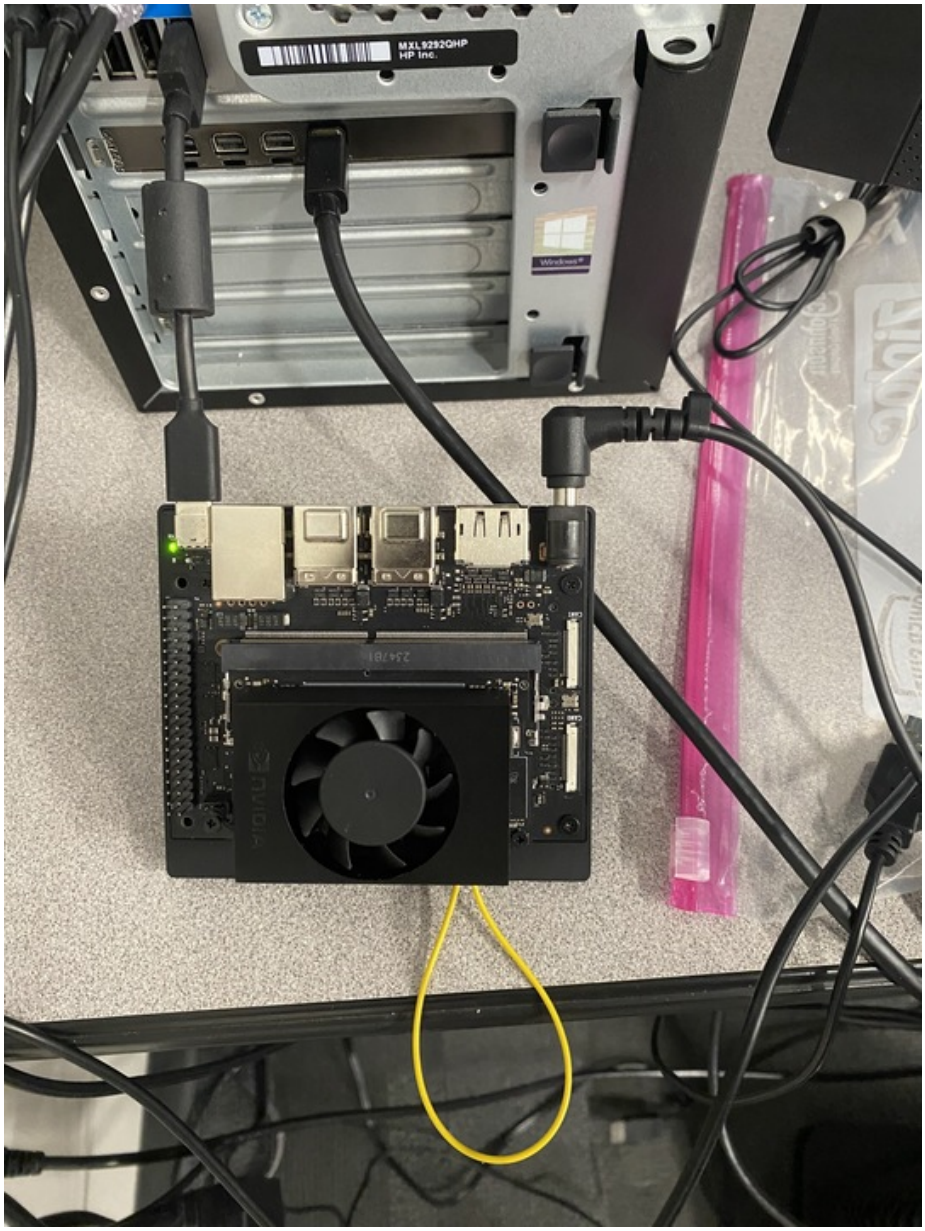
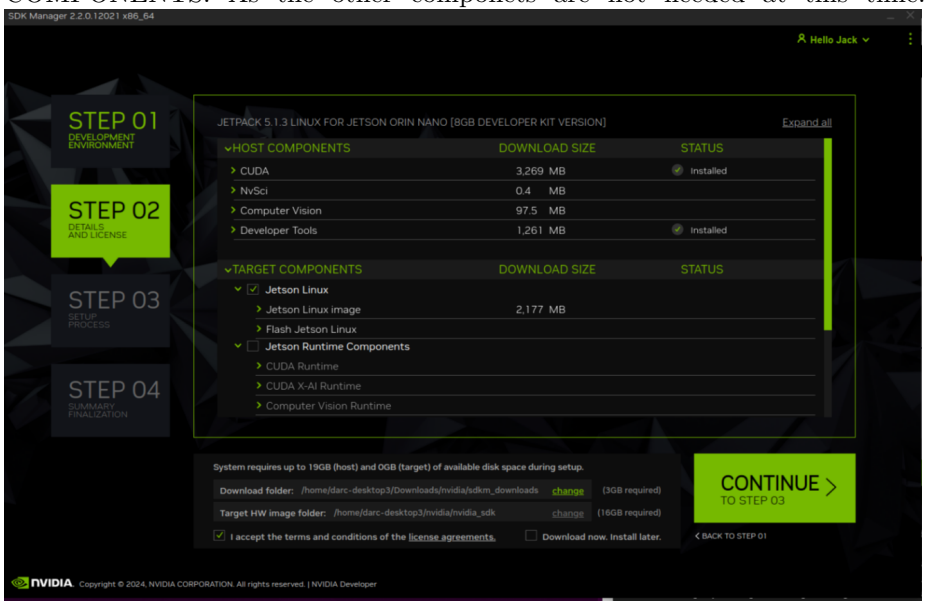


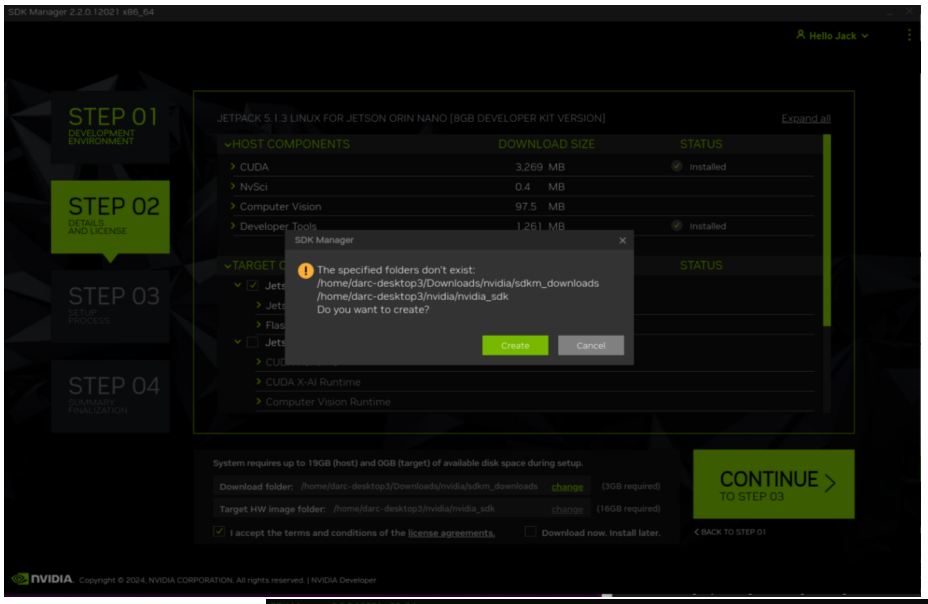
Figure 7: alt text

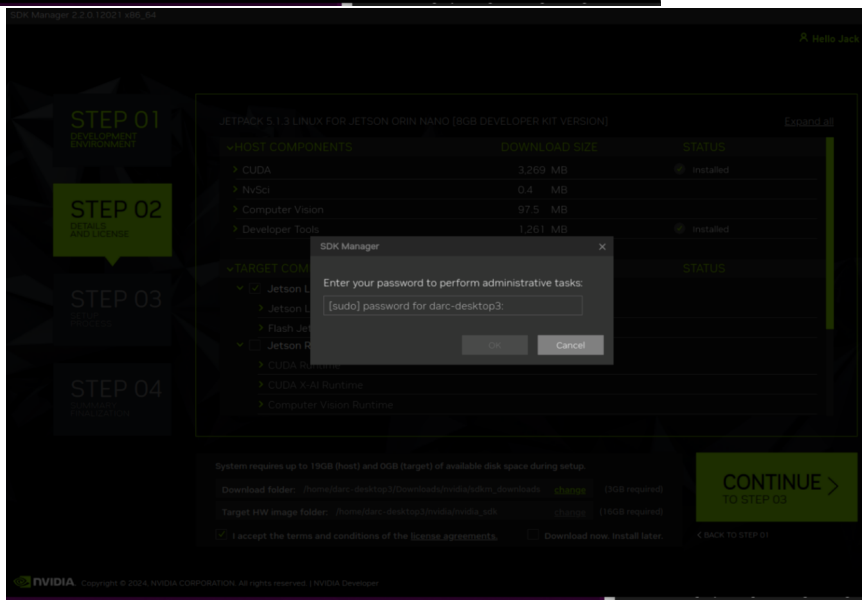
2. Continue to step 2 and **ONLY SELECT** Jetson Linux under TARGET COMPONENTS. As the other componets are not needed at this time.



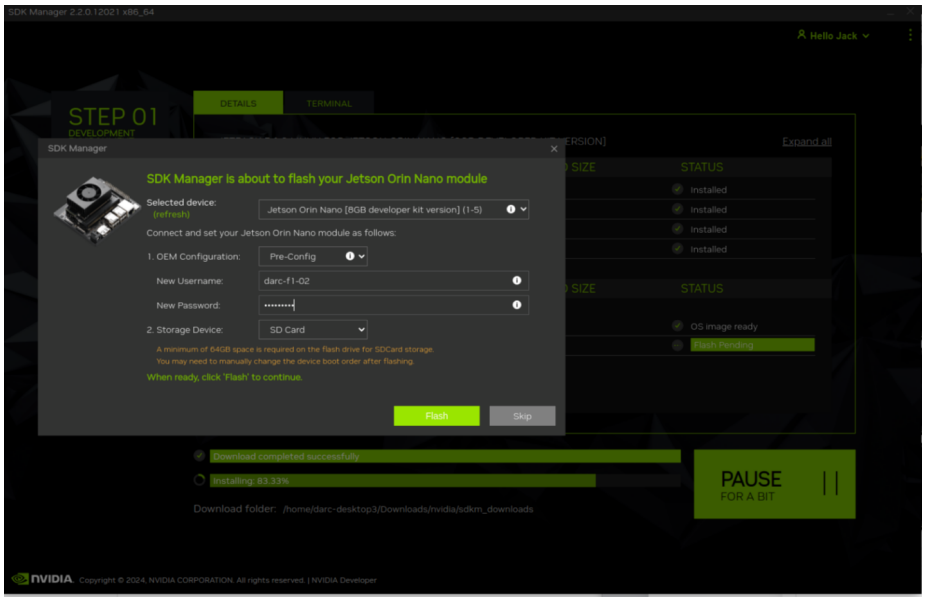
3. Before proceeding to step 3 **DISCONNECT** the jumper wire taking the Jetson out of force recovery mode. 4. It will state the specified folder doesn't exist.

Click **Create**

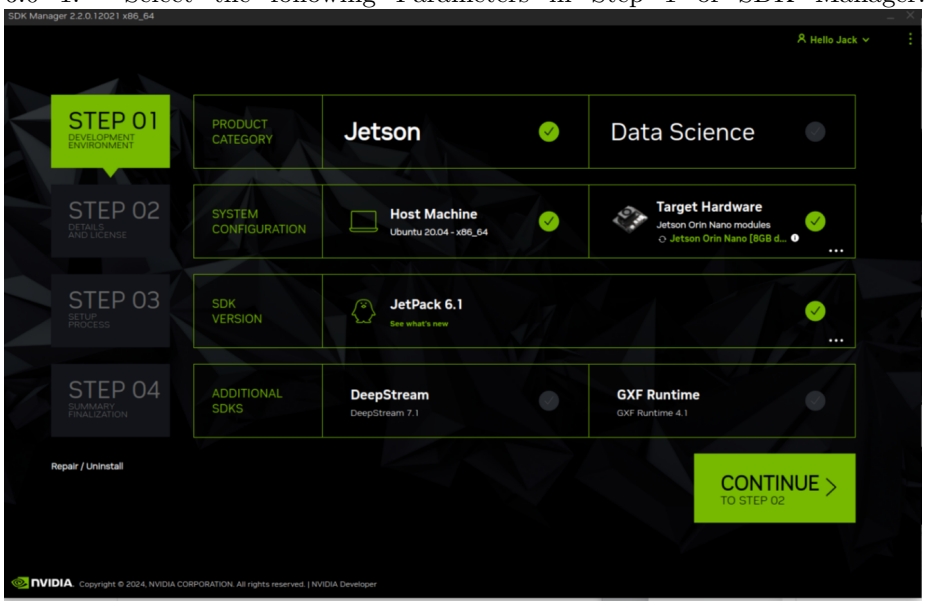




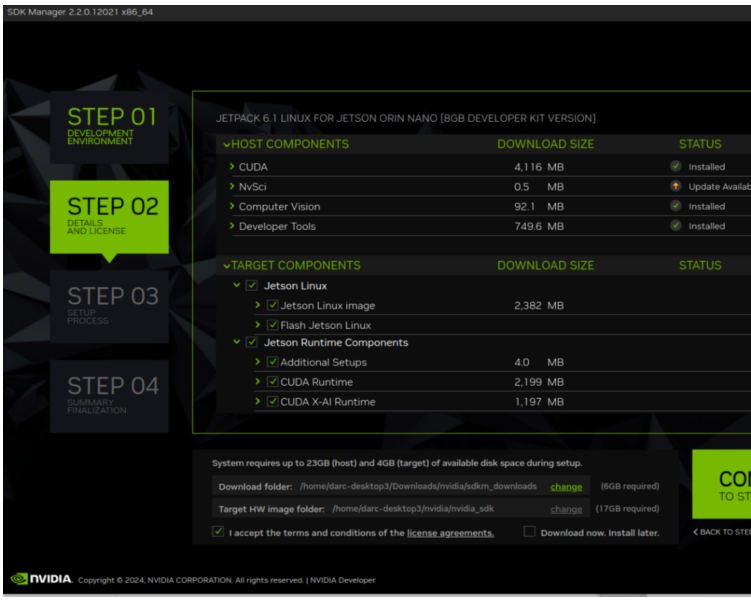
5. Next it will ask for your password. Then all of the host componets will be installed and the Jetson Linux image will be created. 6. Once the OS image is ready the following screen will appear. Select the following parameters shown in the picture.



Once the parameters are selected click flash. ### Flashing Jetpack 6.0 1. Select the following Parameters in Step 1 of SDK Manager.

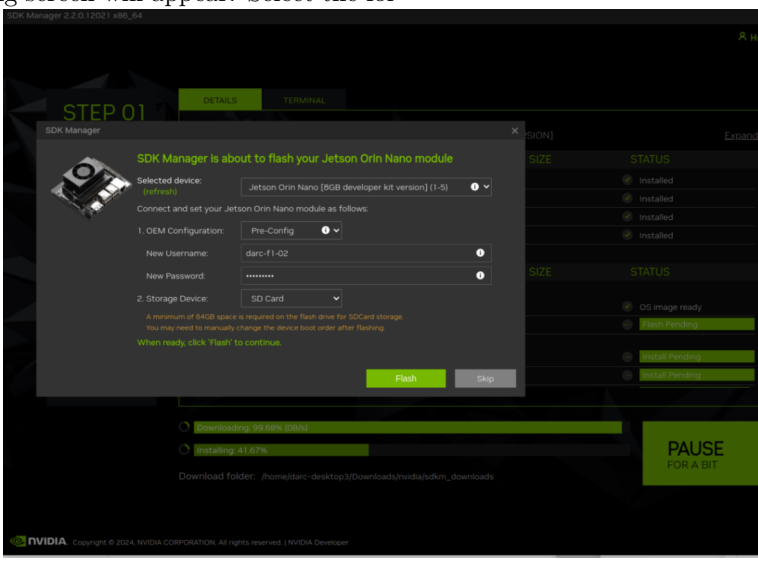


2. Continue to step 2 unsure that Jetson Linux and Jetson RUntime Components



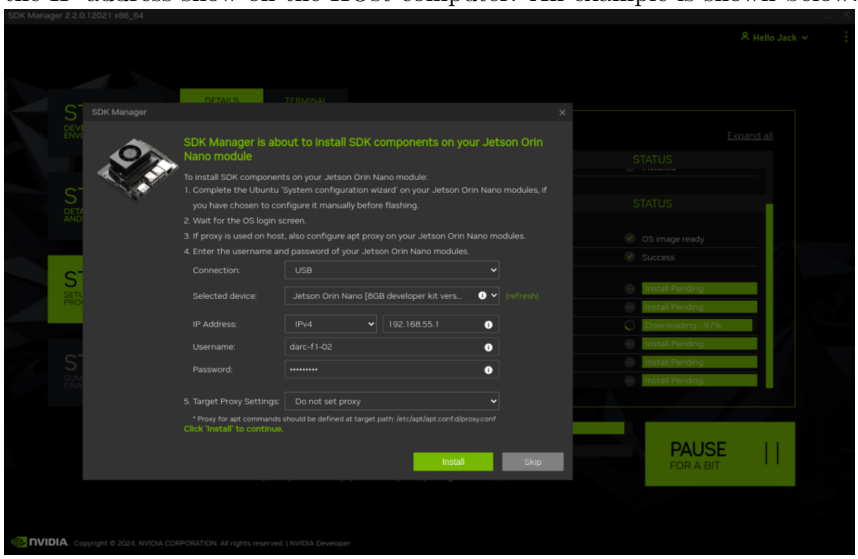
are selected under TARGET COMPONENTS. 3. Before proceeding to step 3 **DISCONNECT** the jumper wire taking the Jetson out of force recovery mode. 4. It will state the specified folder doesn't exist. Click **Create** 5. Next it will ask for your password.

6. Once the OS image is ready the following screen will appear. Select the following parameters shown in the picture. Then all of the host componets will be installed and the Jetson Linux image will be created.



7. Once the the flash finishes connect a monitor via display port, mouse and keyboard. The Jeston should be setup. Then login into the system, open

a terminal and type `ip addr show` confirm that the IP Address matches the IP address show on the HOst computer. An example is shown below.



When this install is complete the Jetson is ready to proceed to the next step.

2. Run Jetson NX from SSD

In the build instruction we applied an SSD NVMe on to the Jetson NX. We will now make use of this SSD by switching the rootfs to point to the SSD. In effect, the system will now run from the SSD, the SD card is only there to boot the system. Therefore everything you install on your system will automatically installed on the SSD.

Please follow this tutorial here that has both video and commands integrated to enable your Jetson NX to run from the SSD

I highly recommend going to the "here" link above it is very helpful even though it is meant for the Jetson Xavier NX

Important

`These script changes the rootfs to the SSD after the kernel image is loaded from the eMMC/SI`

The **here** link will take you to a cite and execute the following steps

1. Ensure the SSD is correctly installed
2. Watch video on how to format SSD using this link
3. Open a terminal window on the Jetson and run the following commands

Note : *You should do this process directly after creating a new SD card.* On the JetsonHacks account on Github, there is a repository rootOnNVMe. Clone the

repository:

```
git clone https://github.com/jetsonhacks/rootOnNVMe
```

and switch over to that repository's directory:

```
cd rootOnNVMe
```

Next, copy the rootfs of the eMMC/SD card to the SSD

```
./copy-rootfs-ssd.sh
```

Finally, we will add a service which will run a script when the system starts up. The script will "pivot the root" to the SSD so that the system will run from the SSD.

```
./setup-service.sh
```

3. Configuring WiFi and SSH

Connect to Wifi

Go to top right corner and select network

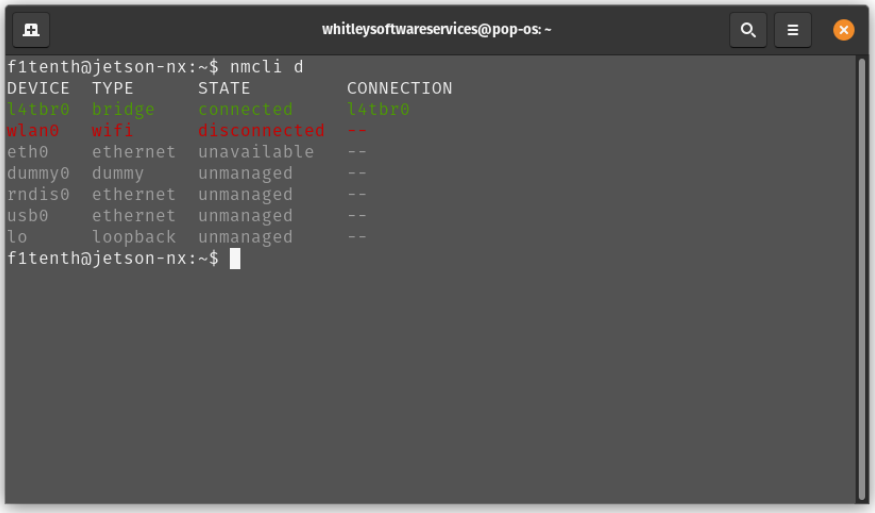
Note: If you are using Labnet you will need to add the MAC address to the Network. To find the MAC address use the command `ip link show`

Here is an example.

Configure SSH

Setting static IP Address

Start by finding the wifi connection device. This can be down by running `nmcli d`



then use the device name for wifi.

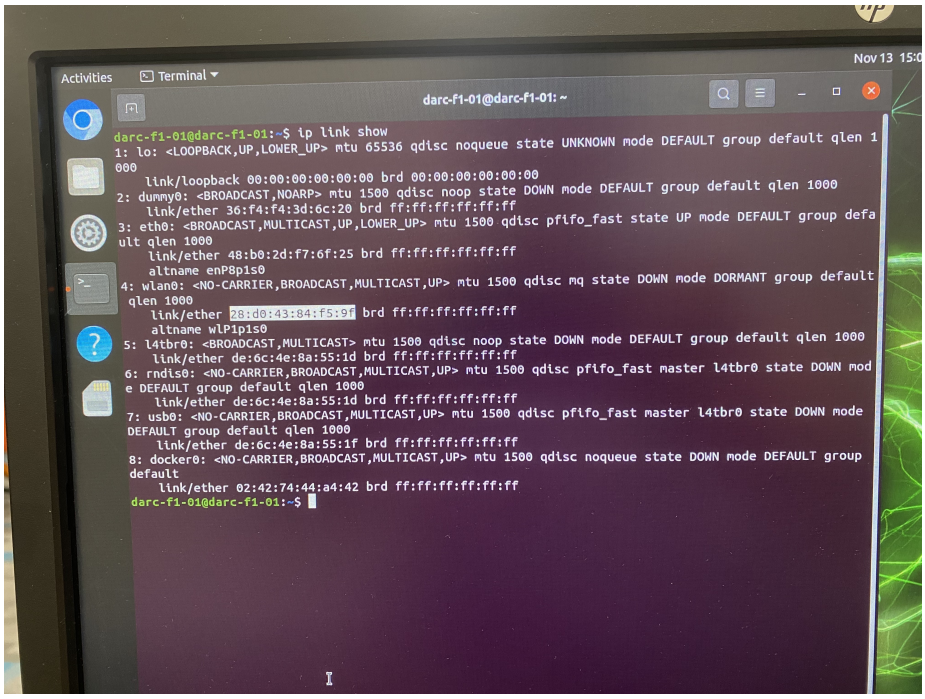
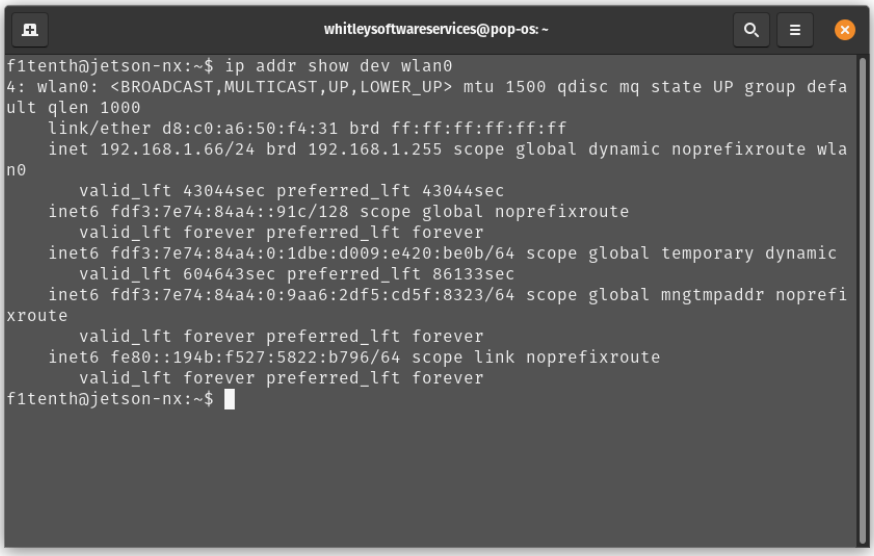


Figure 8: alt text

For the example above the device name is wlan0

1. To get the currently-assigned IP address use the command `ip addr show dev wlan0`.



Currently-connected WiFi IP address.¶

2. To set a static IP address, you will also need to know the name of the connection. This is usually the same as the SSID of the WiFi network but not always. To see the list of current connections, use the command `nmcli c show`.

List of connections.¶

3. To set a static IP address use the command sudo `nmcli c mod [CONNECTION_NAME] ipv4.address [NEW_ADDRESS]/[CIDR]` where `[CONNECTION_NAME]` is replaced with the name of your WiFi connection that you got from step 8, `[NEW_ADDRESS]` is replaced with the static IP address that you want to set, and `[CIDR]` is the CIDR representation of the subnet (usually 24).

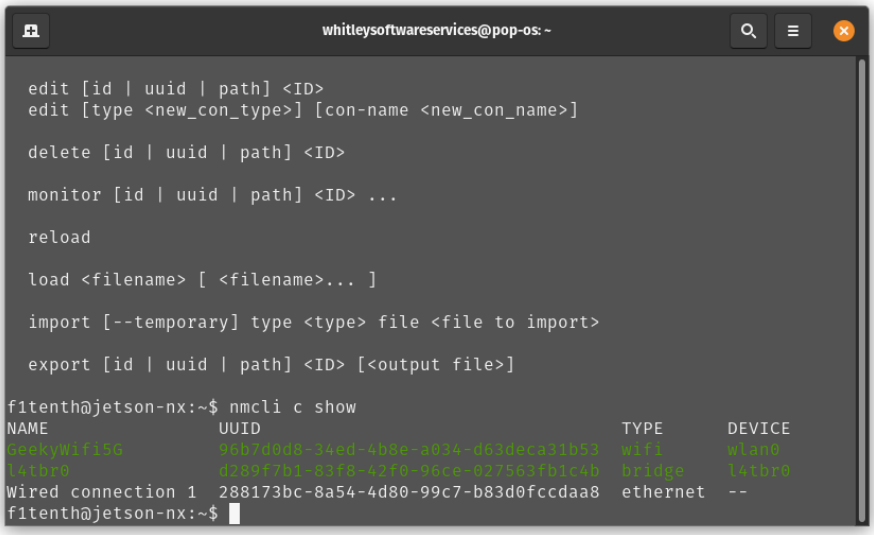
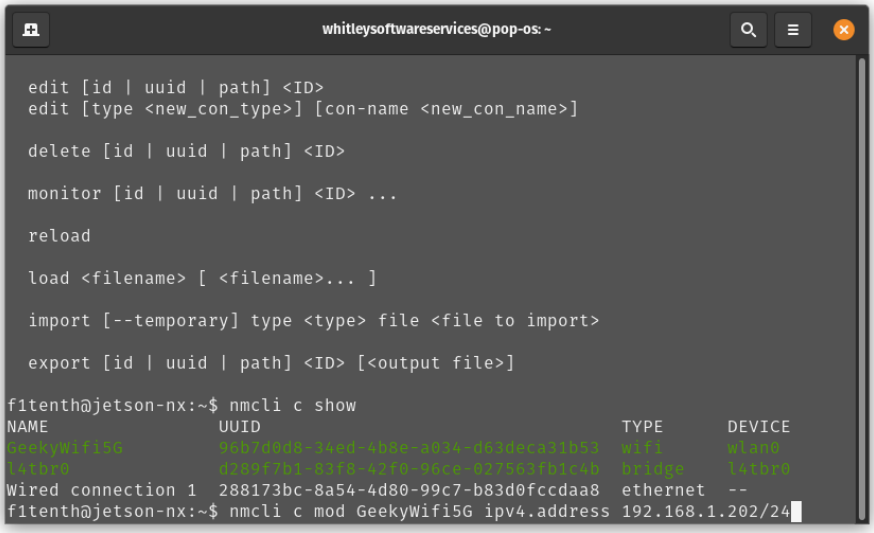


Figure 9: alt text



Setting static IP address.¶

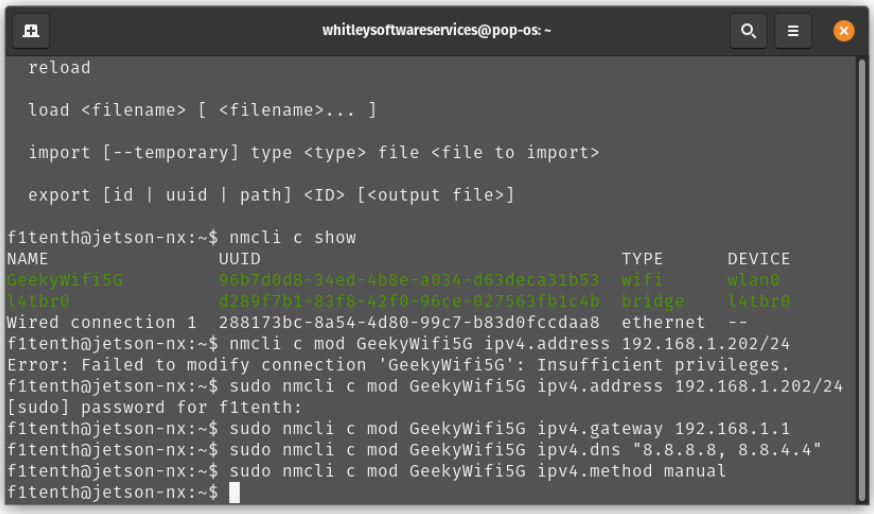
4. To set the connection's default gateway, use the command `sudo nmcli c mod [CONNECTION_NAME] ipv4.gateway [GATEWAY_IP]` where `[CONNECTION_NAME]` is replaced with the name of your WiFi connection that you got from step 8 and `[GATEWAY_IP]` is replaced with the IP

address of your WiFi network's gateway/router.

Setting IP gateway.¶

5. To set the connection's DNS servers, use the command sudo `nmcli c mod [CONNECTION_NAME] ipv4.dns "[DNS_SERVER1]"` where `[CONNECTION_NAME]` is replaced with the name of your WiFi connection that you got from step 8 and `[DNS_SERVERS]` is replaced with a comma-separated list of DNS server IP addresses. Google DNS servers at 8.8.8.8 and 8.8.4.4 are recommended.

6. To disable DHCP and always use the static IP address on this connection, use the command `sudo nmcli c mod [CONNECTION_NAME] ipv4.method manual` where `[CONNECTION_NAME]` is replaced with the name of your WiFi connection that you got from step 8.



Setting connection to always use static IP.¶

7. To save the changes you've made, run the command `sudo nmcli c up [CONNECTION_NAME]` where `[CONNECTION_NAME]` is replaced with the name of your WiFi connection that you got from step 8.

8. To verify that you can SSH into the NVIDIA Jetson Orin Nano Developer Kit, verify that the Pit/Host PC is connected to the same network as the Jetson Orin Nano Developer Kit and use an SSH client on the Host PC to connect to the new IP address of the Developer Kit. On Linux this would be done with the command ssh f1tenth@[IP_ADDRESS] where [IP_ADDRESS] is replaced with the static IP address that you assigned to the Developer Kit. After you have verified that SSH works correctly, you can close the connection to the Developer Kit in your terminal emulator. **After setup SSH using following command**

`ssh -p 22 User@ip adress`

example: `ssh -p 22 darc-f1@10.152.69.100`

Note: If network is having difficulty connecting got to the wi-fi settings under the`IPv4` tab and select `Automatic` unsure the `DNS` is clear.

4. Updating Packages

All further steps assume that your NVIDIA Jetson Orin Nano Developer Kit is connected to the internet. You can execute all the commands directly in the terminal application of the NVIDIA Jetson. Now we are updating the Ubuntu system on the Jetson NX.

1. To update the list of available packages, run `sudo apt update`.

2. To install all available updates, run `sudo apt full-upgrade`.
3. Once all packages have been upgraded run `sudo reboot` to restart the Developer Kit and apply any changes.

5. Setup Bluetooth and Connect Controller

- Steps to Setup Bluetooth on Jetson Orin Nano: Update System Packages: Make sure the system is up to date.

```
sudo apt update
sudo apt upgrade
```

- Install Bluetooth Packages: Install the necessary Bluetooth packages if they're not already installed.

```
sudo apt install bluez bluez-tools
sudo apt install pulseaudio pulseaudio-module-bluetooth
```

- Enable Bluetooth Service: Start and enable the Bluetooth service.

```
sudo systemctl start bluetooth
sudo systemctl enable bluetooth
```

If your adapter is detected, you should see output like hci0: Type: BR/EDR .... If no device is found, it might be due to a missing driver or unsupported hardware.

Check if bluez and bluez-tools are installed: Run this command:

```
sudo dpkg -l | grep bluez
```

If the packages are installed, you should see an output like this:

```
ii  bluez          <version>     <architecture>   <description>
ii  bluez-tools    <version>     <architecture>   <description>
```

Run this command:

```
dpkg -l | grep pulseaudio
```

This will show all `pulseaudio` related packages that are installed. Look for `pulseaudio` and `pulseaudio-module-bluetooth` in the output. If installed, the output will look similar to:

```
ii  pulseaudio                      <version>     <architecture>   <description>
ii  pulseaudio-module-bluetooth     <version>     <architecture>   <description>
```

If they are not installed, there will be no output or only partial results.

Now you should be able to go to settings and connect the controller

6. Creating a Swapfile

1. Run the following commands to create a swapfile which can help with memory-intensive tasks

```
sudo fallocate -l 4G /var/swapfile
sudo chmod 600 /var/swapfile
sudo mkswap /var/swapfile
sudo swapon /var/swapfile
sudo bash -c 'echo "/var/swapfile swap swap defaults 0 0" >> /etc/fstab'
```

2. Connecting the Pit/Host and the Jetson Orin Nano

Overview

We could now log into the Jetson using a monitor, keyboard, and mouse, but ideally we would want remote access when we're driving the car. Throughout this tutorial, you will be asked to configure the Jetson's and your laptop's network settings. Make sure to get these right! Using the wrong IP address may lead to conflicts with another classmate, meaning neither of you will be able to connect.

If your **Pit/Host** computer has WiFi capability, you connect both the computer and the F1TENTH car to a wireless router which reserves a static IP address for Jetson Orin Nano on the vehicle.

If the **Pit/Host** computer doesn't have Wifi capability:

1. Connect the Pit/Host computer to a WiFi router via an ethernet cable.
2. Connect the NVIDIA Jetson NX to the same router via Wifi.

To make this section easy to follow, the Routers WiFi network SSID will be called and referred to `F1TENTH_WIFI`. In your scenario, it'll be the SSID of your router's access point.

1. Vehicle Hardware Setup

If you have a NVIDIA Jetson Orin Nano, it comes with a network card onboard. Make sure the antennas are connected. The battery should be plugged into the vehicle and the Powerboard should be on.

2. Connecting the NVIDIA Jetson NX to WiFi

Power up the F1TENTH vehicle and connect the car to a monitor (via HDMI) and both a mouse and keyboard (via USB). You see the Jetson Orin Nano is showing its main Desktop which is an UBUNTU 22.04 version. You can now connect the NVIDIA Jetson Orin Nano to the Labnet by clicking on wireless

icon on top-right corner of Ubuntu Desktop and selecting Labnet. It might take a while for the NVIDIA Jetson Orin Nano to discover the wireless network.

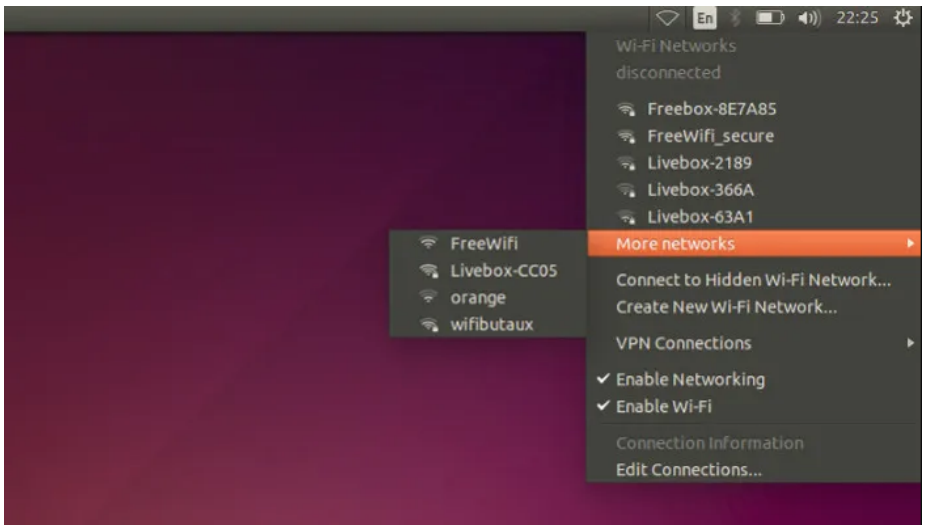


Figure 10: alt text

After you're connected to the wireless network, open a terminal and type:

`You should see something similar to this:`

eth0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500 ether 00:04:4b:cb:d9:52 txqueuelen 1000 (Ethernet) RX packets 0 bytes 0 (0.0 B) RX errors 0 dropped 0 overruns 0 frame 0 TX packets 0 bytes 0 (0.0 B) TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0 device interrupt 40

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536 inet 127.0.0.1 netmask 255.0.0.0 inet6 ::1 prefixlen 128 scopeid 0x10 loop txqueuelen 1 (Local Loopback) RX packets 1047 bytes 82631 (82.6 KB) RX errors 0 dropped 0 overruns 0 frame 0 TX packets 1047 bytes 82631 (82.6 KB) TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

rndis0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500 ether ea:a2:37:fd:d0:e1 txqueuelen 1000 (Ethernet) RX packets 0 bytes 0 (0.0 B) RX errors 0 dropped 0 overruns 0 frame 0 TX packets 0 bytes 0 (0.0 B) TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

usb0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500 ether ea:a2:37:fd:d0:e3 txqueuelen 1000 (Ethernet) RX packets 0 bytes 0 (0.0 B) RX errors 0 dropped 0 overruns 0 frame 0 TX packets 0 bytes 0 (0.0 B) TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlan0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500 inet 195.0.0.5 netmask 255.255.255.0 broadcast 195.0.0.255 inet6 fe80::4df8:b83b:9390:319d

prefixlen 64 scopeid 0x20 ether 0c:dd:24:99:e6:52 txqueuelen 1000 (Ethernet) RX packets 12511 bytes 4918686 (4.9 MB) RX errors 0 dropped 0 overruns 0 frame 0 TX packets 1262 bytes 196668 (196.6 KB) TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0 NoMachine Ctrl+alt+d‘

```
You should be able to find your car's assigned IP address under ```wlan0```, then after inet

## 3. Connecting the Pit/Host Computer to WiFi

Now, on the Pit/Host laptop, connect to the same wireless network, ```Labnet``` and find its

If you're running Linux on the Pit laptop in a virtual machine (VM), connect the Pit compute

## 4. Connecting to the Pit/Host to the NVIDIA Jetson NX

Now that the car and the laptop are on the same network, you should check that you can ping

On the NVIDIA Jetson NX, open a terminal and type: ```ping 192.168.1.151``` (This is the IP
On the Pit computer, open a terminal and type ```ping 195.0.0.5``` (This is the IP address o
Remember to replace the IP addresses in the two lines above with your specific addresses.

You can now SSH into your car from your laptop. Use ``ssh`` in the terminal if you're on [ma

**An example of this ssh command is as follows - ```ssh -p 22 darc-f1-01@10.152.69.216```**

We recommend using [tmux](https://www.hamvocke.com/blog/a-quick-and-easy-guide-to-tmux/)  wh

<!--  # 5. Using a Remote Desktop

### I am not currently using the remote desktop I am only utilizing ssh for Host Machine

Although we now have SSH access to the car, it is still inconvenient to run GUI application

First, download NoMachine for your **pit/host** computer's specific OS [here](https://www.no

After NoMachine is installed on both sides, go to your pit/host's NoMachine, click **Add** t

#### A. Install [NoMachine](https://www.nomachine.com/download) on Host/Pit computer
#### B. Download and install NoMachine on Jetson with the following commands
```wget https://www.nomachine.com/free/arm/v8/deb -O nomachine.deb```

```sudo dpkg -i nomachine.deb```
#### C. Setup GNOME Flashback instead of XFCE using NoMachine

Run the following code in terminal
```

```sudo apt install gnome-session```

```sudo systemctl set-default multi-user.target```

```sudo gedit /usr/NX/etc/node.cfg```

Replace the following DefaultDesktopCommand with ```DefaultDesktopCommand "/usr/bin/gnome-se

Run the following line in the terminal

```sudo /usr/NX/bin/nxserver --restart```

```sudo reboot```
-->

Install F1TENTH Driver Stack

Note -
This section assumes that you have already completed Building the Car and System Configurati

At the end of this section, you will have the VESC tuned and the lidar connection completed.

Difficulty Level: Intermediate-Advanced

Approximate Time Investment: 1.5 hour

With the physical car is built and the system configuration setup, we can start to install t

1. Configuring the VESC

Important Safety Tips
Put your car on an elevated stand so that its wheels can turn without it going anywhere. If

Make sure you hold on to the car while testing the motor to prevent it from flying off the s

Make sure there are no objects (or people) in the vicinity of the wheels while testing.

Use a fully-charged LiPO battery instead of a power supply to ensure the motor has enough cu

1. Installing the VESC Tool
We need to configure the VESC so that it works with our motor and vehicle transmission. Befo

2. Powering the VESC
First we need to power the VESC. Plug the battery in, and make sure the polarity is correct.

![alt text](<Pictures/F1tenth Build/vesc01.jpeg>)
```

Next, unplug the USB cable of the VESC from the Jetson NX and plug the USB into your laptop

3. Connecting the VESC to Your Laptop
Launch the VESC Tool. On the Welcome page, press the AutoConnect button on bottom left of th

Use AutoConnect

4. Updating the Firmware on the VESC
The first thing you'll need to do is to update the firmware onboard the VESC. Depending on t

With VESC Tool versions released after Mar. 31 2021, you can use the latest default firmware

5. Uploading the Motor Configuration XML
After firmware update, Select Load Motor Configuration XML from the drop down menu and selec

6. Detecting and Calculating Motor Parameters
To detect and calculate the FOC motor parameters, navigate to the FOC tab under Motor Settin

After the motor parameters are measured, the fields at the bottom of the screen should turn

7. Changing the Openloop Hysteresis and Openloop Time

Navigate to the **Sensorless** tab on top of the screen. Change the **Openloop Hysteresis**

8. Tuning the PID controller
Now you can start tuning the speed PID controller. To see the RPM response from the motor, r

To create a step response for the motor, you can set a target RPM at the bottom of the scree

You want to look for a clean step response that has a quick rise time and zero to very littl

These are my settings.

9. Changing the hardware speed limit
By default, the motor configuration sets a safe top motor RPM. If you wish to change the har

Danger

Please see the Odometry Tuning section in the software stack setup to see how vehicle velocity is converted to ERPM for the motor to calculate a safe maximum erpm for your motor.

2. Hokuyo 10LX Ethernet

Note

If you have a 30LX or a LIDAR that connects via USB, you can skip this section.

Approximate Time Investment: 30 minutes

Connect to the Jetson NX either via SSH or a wired connection (monitor, keyboard, mouse).

In order to utilize the 10LX you must first configure the eth0 network. From the factory the

Open Network Configuration in the Linux GUI on the Jetson Orin Nano. In the ipv4 tab, add a

IP address ```192.168.0.15```

Subnet mask is ```255.255.255.0```

Gateway is ```192.168.0.10```

Call the connection Hokuyo. Save the connection and close the network configuration GUI.

When you plug in the 10LX make sure that the Hokuyo connection is selected. If everything is

If thisn doesn't work use the following commands

Open the terminal

sudo ip addr add 192.168.0.1/24 dev eth0

Then go to setting and edit the eth0 connection. By going to the ipv4 tab and entering in th

Now try to ping ```192.168.0.10```.

3. F1TENTH Driver Stack Setup

NOTE: This is how to setup the driver stack WITHOUT DOCKER

Approximate Time Investment: 1.5 hour

Overview

This setup is being ran on JetPack 6.0 since the Jetson can run on Ubuntu 22.04, and we can

In the following section, we'll go over how to set up the drivers for sensors and the motor

A. Setting up udev rules for our sensors.

B. Installing ROS 2 and its utilities.

C. Setting up the driver stack.

D. Launch teleoperation and the LiDAR.

Everything in this section is done on the **Jetson Orin Nano Dev Kit** so you will need to c

A. udev Rules Setup

*Goal is to name the VESC and the Hokuyo Lidar (Ethernet connection)

Naming the Lidar Sensor

```sudo gedit /etc/udev/rules.d/99-ethernet-hokuyo.rules```

```SUBSYSTEM=="net", ACTION=="add", ATTR{address}=="48:b0:2d:eb:e4:dd", NAME="hokuyo"```

Naming the VESC

```sudo gedit /etc/udev/rules.d/99-vesc.rules```

```KERNEL=="ttyACM[0-9]*", ACTION=="add", ATTRS{idVendor}=="0483", ATTRS{idProduct}=="5740",

Finally, trigger (activate) the rules by running

sudo udevadm control –reload-rules sudo udevadm trigger

Reboot your system, and you should find VESC by running ```ls /dev/sensors```.

## B. Installing ROS 2 and its Utilities

Below are instructions on how to install ROS 2 humble and the necessary utilites.

### Set locale
Make sure you have a locale which supports UTF-8. If you are in a minimal environment (such

locale # check for UTF-8

sudo apt update && sudo apt install locales sudo locale-gen en_US en_US.UTF-8 sudo update-locale LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8 export LANG=en_US.UTF-8

locale # verify settings

### Setup Sources
You will need to add the ROS 2 apt repository to your system.

First ensure that the Ubuntu Universe repository is enabled.

sudo apt install software-properties-common sudo add-apt-repository universe

Now add the ROS 2 GPG key with apt.

sudo apt update && sudo apt install curl -y sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistro/maste -o /usr/share/keyrings/ros-archive-keyring.gpg

Then add the repository to your sources list.

echo “deb [arch=$(dpkg –print-architecture) signed-by=/usr/share/keyrings/ros-archive-keyring.gpg] http://packages.ros.org/ros2/ubuntu $(. /etc/os-release && echo $UBUNTU_CODENAME) main” | sudo tee /etc/apt/sources.list.d/ros2.list > /dev/null

### Install ROS 2 packages
Update your apt repository caches after setting up the repositories.

sudo apt update

ROS 2 packages are built on frequently updated Ubuntu systems. It is always recommended that

sudo apt upgrade

### Warning

Due to early updates in Ubuntu 22.04 it is important that systemd and udev-related packages

Please refer to ros2/ros2#1272 and Launchpad #1974196 for more information.

Desktop Install (Recommended): ROS, RViz, demos, tutorials.

sudo apt install ros-humble-desktop

### Try an Example to see if it works

In one terminal, source the setup file and then run a C++ talker:

source /opt/ros/humble/setup.bash ros2 run demo_nodes_cpp talker

In another terminal source the setup file and then run a Python listener:

source /opt/ros/humble/setup.bash ros2 run demo_nodes_py listener

You should see the `talker` saying that it's `Publishing` messages and the `listener` saying

### Now install ROS Utilities

### Install Colcon

sudo apt install python3-colcon-common-extensions

### Install dependencies

sudo apt install python3-bloom python3-rosdep fakeroot debhelper dh-python

### Initialize rosdep

Initialize the rosdep database by calling:

sudo rosdep init rosdep update

Note that the `rosdep init` command may fail if it has already been initialized in the past;

## D. Setting up the driver stack

First, we'll create a ROS 2 workspace for our driver stack with the following commands. We'l

cd $HOME mkdir -p f1tenth_ws/src

Then, make this into a ROS 2 workspace by running:

cd f1tenth_ws colcon build

Next, we'll clone the repo into the src directory of our workspace:

cd src git clone https://github.com/f1tenth/f1tenth_system.git

Then use the following commands to use the humble-devel branch from the F1tenth repo

cd f1tenth_system git checkout humble-devel

Then we'll update the git submodules and pull in all the necessary packages

git submodule update –init –force –remote

After git finishes cloning, we can now install all dependencies for our packages with rosdep

cd $HOME/f1tenth_ws source /opt/ros/humble/setup.bash rosdep update rosdep install –from-paths src -i -y

Lastly, after dependencies are installed, we can build our workspace again with the driver s

colcon build

You can find more details on how the drivers are set up in the README of the f1tenth_system
Install colcon If you haven't installed colcon yet, follow these steps:
### Error

If the following error occurs

CMake Error at /opt/ros/humble/share/io_context/cmake/io_context-extras.cmake:17 (find_package): By not providing "Findasio_cmake_module.cmake" in CMAKE_MODULE_PATH this project has asked CMake to find a package configuration file provided by "asio_cmake_module", but CMake did not find one.

Could not find a package configuration file provided by "asio_cmake_module" with any of the following names:

asio_cmake_moduleConfig.cmake
asio_cmake_module-config.cmake

Add the installation prefix of "asio_cmake_module" to CMAKE_PREFIX_PATH or set "asio_cmake_module_DIR" to a directory containing one of the above files. If "asio_cmake_module" provides a separate development package or SDK, be sure it has been installed. Call Stack (most recent call first): /opt/ros/humble/share/io_context/cmake/io_contextConfig.cmake:41 (include) /opt/ros/humble/share/serial_driver/cmake/ament_cmake_export_dependencies-extras.cmake:21 (find_package) /opt/ros/humble/share/serial_driver/cmake/serial_driverConfig.cmake:41 (include) /opt/ros/humble/share/ament_cmake_auto/cmake/ament_auto_find_build_dependencies.cmake:67 (find_package) CMakeLists.txt:14 (ament_auto_find_build_dependencies)

Use the following code

sudo apt update sudo apt install ros-humble-asio-cmake-module

**There will be several warnings please use the following commands to midigate these warning

Code format: `find . -type f -exec sed -i 's/old/new/g' {} \;`

find . -type f -exec sed -i ‘s/script-dir/script_dir/g’ {} ; find . -type f -exec sed -i ‘s/install-script/install_script/g’ {} ;

## E. Launch teleoperation and the LiDAR

This section assumes that the lidar has already been plugged in (either to the USB hub or to

Before the bringup launch, you’ll have to set the correct parameters according to which LiDA

$HOME/f1tenth_ws/src/f1tenth_system/f1tenth_stack/config/

If you’re using an ethernet based LiDAR, set the ip_address field to the corresponding ip ad

If you’re using a USB based LiDAR, comment out the ip_address field, and uncomment the line

In your running container, run the following commands to source the ROS 2 underlay and our w

source /opt/ros/humble/setup.bash cd $HOME/f1tenth_ws source install/setup.bash

Then, you can launch the bring up with:

ros2 launch f1tenth_stack bringup_launch.py

Running the bringup launch will start the VESC drivers, the LiDAR drivers, the joystick driv

source /opt/ros/humble/setup.bash cd $HOME/f1tenth_ws source install/setup.bash rviz2

Change the **Fixed Frame** parameter from **map** to **laser**
The rviz window should show up. Then you can add a LaserScan visualization in rviz on the /s

If RViz2 is populating a blank screen is could be because RViz2 is misconfigured. Try the fo

1. Close RViz2
2. Delete the saved configuration file (if it exists):

rm ~/ .rviz2/default.rviz

3. Relaunch RViz2:

rviz2

# Test controller

connect the controller using bluetooth from the settings

sudo apt install jstest-gtk

Then run `jstest-gtk` to open the tester and by selecting the3 wireless controller

```
![alt text](image-7.png)

Manual Test of Teleop Commands
If you confirm that joystick inputs are present but the car isn't responding, you can try pu
```

ros2 topic pub /ackermann_cmd ackermann_msgs/msg/AckermannDriveStamped
“drive: {speed: 1.0, steering_angle: 0.0}”

```
# Driving the F1Tenth Car

## Manual Driving

### Overview
Before we can get the car to drive itself, it’s a good idea to test the car to make sure it

You **MUST** connect to the Jetson via SSH or Remote Desktop for this section.

### 1. Vehicle Inspection
We want to minimize the number of accidents so before we begin, let’s first inspect our vehi

1. Make sure you have the car running off its LIPO battery.

2. Plug the USB dongle receiver of the Logitech Joypad into the USB hub.

3. Make sure you have the VESC connected.

4. Ensure that both your car and laptop are connected to a wireless access point if you neec

5. Make sure you’ve cloned the f110_system repository and set up your docker container as ex

6. This section uses the program tmux (available via apt-get) to let you run multiple termir

### 2. Driving the Car

1. Open a terminal on the Pit laptop and SSH into the car from your computer.

2. Launch teleop following the instructions for Launching and Testing teleop and the LiDAR i

3. Hold the LB button (Dead man’s switch) on the controller to start controlling the car. Us

### Troubleshooting
During teleop, if the joystick is not mapped correctly, you can change the mapping in `/f1te

If **nothing happens**, one reason can be that the driver name is listening on the wrong por

Note that the **LB button acts as a "dead man’s switch"**, as releasing it will stop the car
```

You can see a **mapping of all controls** used by the car in the `joy_teleop.yaml` file. For

Motor rotation direction negated. If you're car is driving backwards when commanded driv

VESC out of sync errors: Check that the VESC is connected. If the error persists, make s

Serial port busy errors: Your VESC might have just booted up, give it a few seconds and

SerialException errors and you're using the 30LX Hokuyo, the errors might be due to a po

urg_node related errors: Check the ports (e.g. an ip address in sensors.yaml can only be

Now is the Time to setup NoMachine

Calibrating the Odometry

Calibrating the steering and odometry

Now that everything is built, configured, and installed, the odometry of the vehicle needs t

1. The parameters in vesc.yaml need to be calibrated. This yaml file is located at:
HOME/f1tenth_ws/src/f1tenth_system/f1tenth_stack/config/vesc.yaml
if you're using the driver stack natively, or on the host if you're using docker containers,
$/f1tenth_ws/src/f1tenth_system/f1tenth_stack/config/vesc.yaml
in the container if you're using docker containers. Note that the config directory has all t

2. In the first few steps, make sure you've lifted the car up with a pit stand or a box, so

3. First, we need to check if our motor is rotating in the right direction. If when given a

4. Next, we'll also need to check if the vesc driver is interpreting the motor rotation dire
$HOME/f1tenth_ws/src/f1tenth_system/vesc/vesc_ackermann/src/vesc_to_odom.cpp
if you're using the driver stack natively, or on the host if you're using docker containers,
/f1tenth_ws/src/f1tenth_system/vesc/vesc_ackermann/src/vesc_to_odom.cpp
in the container if you're using docker containers.

Modify line `100` so it reads:

double current_speed = -(-state->state.speed - speed_to_erpm_offset_) /
speed_to_erpm_gain_; "'

After changing the source code, you'll have to go back to the workspace at /f1tenth_ws, and call colcon build again. Also remember that whenever you rebuild the workspace, you'll have to call the commands to source the underlay and the overlay again.

Note

In the following steps that adjust parameters in the yaml file, you'll have to call `colcon build` after every change before launching teleop again.

5. After the motor directions are fixed, and the odometry is in the right direction, we can now start tuning the steering and the odometry gains. The first to be tuned is the Steering Offset. This parameter will determine the offset we put on the servo, and whether the car can drive straight when given no steering input. Again, start teleop with the bringup launch. Drive the car in a straight line a couple times with no steering input for a couple meters, and see if it's driving straight. Adjust the steering_angle_to_servo_offset parameter in vesc.yaml if it's not. Make small adjustment everytime (in the magnitude of 0.1). Repeat until you find the correct offset so the car drives straight.

Note

In the following steps, you'll need to lay down the tape measure straight on the ground. It is helpful to tape the tape measure so it doesn't move around.

Next, we'll tune the Steering Gain. This parameter determines the smallest turning radius of the car. We'll determine the desired turning radius given the maximum steering angle of the car we're setting, which is 0.36 radians in both directions. The corresponding turning radius could be then calculated with $R = L/(2\sin(\beta))$, where L is the wheelbase of the car, which is around 0.33 meters; β is calculated as $\arctan(0.5\tan(\delta))$, with δ being the maximum steering angle of the car. After calculations, when turning a half circle, the desired diameter of the half circle should be 1.784 meters.

Place the car at the 0 of the tape measure such that the rear axle of the car is parallel to the tape measure, and the x-axis (roll axis) of the car coincides with the tape measure at 0.

Start teleop. Set the steering angle to the maximum to one side depending on your setup (e.g. if the rest of the tape measure is on the left side of the car, turn left).

Hold the steering, and drive forward slowly and steadily until the car runs over the tape measure again and the rear axle realigns with the tape measure. Now the car should be in the opposite direction to where you started.

Take a measurement on the tape measure. The goal is 1.784 meters. If the measurement overshoots, increase the gain slightly (0.1 at a time). If it undershoots,

decrease the gain. Repeat the process until you've hit 1.784 meters.

If you notice that the wheels turn to different angles on the two directions when give maximum steering angles, adjust the servo_min and servo_max numbers until they're symmetric.

Next, we'll tune the ERPM gain. This parameter determines the conversion from desired velocity in meters/second to desired motor ERPM.

Place the car at the 0 of the tape measure such that the rear axle of the car coincides with 0 and the x-axis (roll axis) of the car is parallel to the tape measure.

Start teleop. Open another bash window in the container, and run ros2 topic echo –no-arr /odom. We're particularly interested in the pose/pose/position/x value. Before giving any driving commands on the joystick, this value should be zero. If it is not zero, kill teleop and restart teleop.

If you notice this value is drifting even when the car is stationary. Change the speed_to_erpm_offset value so that it stops drifting.

Drive slowly and steadily forward without any steering input for more than 3 meters. Record the distance traveled by the car on the tape measure. Note that you'll have to take the reading from the rear axle.

Compare the measured value to the value shown in the echoed message. If the distance reported by echo is larger, decrease the speed_to_erpm_gain value. Otherwise increase the gain. The change is usually on the order of thousands. Note that changing this value also changes the forward speed of teleop. Please drive carefully once the velocity is calibrated. If the forward speed when teleoping is too high, change the scale in human_control for drive-speed in joy_teleop.yaml.

Repeat the process until these values are within 2-3 cm.

Changing the software speed limit If you wish to change the top speed of the car and has already followed the instructions to change the hardware limit in the vesc firmware section. All you'll need to do is also change the speed_min and speed_max values in vesc.yaml. Note that the corresponding max speed in meters/second will be the max erpm value divided by the erpm gain. (e.g. speed_max/speed_to_erpm_gain)

Tip

If you have any build and/or setup questions, post to the forum.